REMARKS

In a final office action dated December 21, 2004, the Examiner rejected claims 1, 2, 4-6, 8, 9, 11-13, 15, 16 and 18-20 under 35 U.S.C. §112, first paragraph, as failing to comply with the enablement requirement[1], rejected claims 1-4, 6, 8-11, 13, 15-18 and 20 under 35 U.S.C. §103(a) as obvious over Cierniak, et al., "Briki: an Optimizing Java Compiler" in view of Hastings (US Patent 5,835,701), and rejected claims 5, 12 and 19 under 35 U.S.C. §103(a) as obvious over *Cierniak* and *Hastings*, and further in view of Wolczko, et al. (US Patent 6,115,782). Claims 7, 14 and 21 were allowed.

Enablement rejections

Applicants respectfully traverse the Examiner's enablement rejections.

In response to the first office action herein, applicants amended certain independent claims. The office action had rejected these claims based on art disclosing a control mechanism for an array of hardware antifuses. Although applicants' claims had recited an "array object", it was clear from applicants' specification that the array objects were a form of data that was manipulated in a computer. Applicants therefore felt that the Examiner's rejection reading "array object" recited in the claims on an array of hardware antifuses was unfounded. However, since they never intended the recited "array object" to encompass an array of hardware devices, applicants were perfectly willing to make explicit what was already clear from any fair reading of

[1] In the office action, at paragraph 3, the Examiner rejects "claims 1-6, 8-13 and 15-20" on these grounds. However, claims 3, 10 and 17 are verbatim recitations of the original claims 3, 10 and 17, re-written in independent form. These claims do not contain the recitations of "digital data objects" and "digital data processing device" objected to by the Examiner. Since the grounds of objection do not apply to claims 3, 10 and 17, and no grounds were offered for these claims, it is assumed that the inclusion of claims 3, 10 and 17 in the list of claims rejected under 35 USC 112 was an unintentional error.

the specification. Applicants therefore amended the independent claims to recite that the "array object" was a "digital data object storable in addressable data storage locations" of a "computer" or a "digital data processing device"[2]

The Examiner objects to the use of the phrases "digital data object" and "digital data processing device", on the grounds that these phrases are not used in the specification. There is no statutory requirement that the precise words used in the specification be used in the claims, provided that what is claimed is disclosed in the specification sufficiently to enable one skilled in the art to make or use the invention.

Regarding the phrase "digital data processing device", the specification discloses that the invention is practiced on a "computer" or multiple computers (See Fig. 4, Spec. p. 9, lines 2-26, etc.) In the context of the specification, this computer is clearly a ***digital computer***, and would be so understood by one of skill in the art.[3] For example, the computer supports the Java virtual machine (Spec. p. 9, lines 5-7) which is well known among those of skill in the art as a program or set of programs for execution on a digital computer. The computer contains an operating system, CPU and memory, all of which are digital computing constructs. The specification repeatedly refers to execution on a processor; in fact, the thrust of the entire invention is computer programming code which executes on a processor. Again these are all constructs in the realm of digital computing. A digital computer is a form of "digital data processing device", and as such the phrase "digital data processing device" is supported by the specification.

---

[2] Independent claim 1 recites "digital data processing device" and claims 8 and 15 recite "computer". A "computer" was recited in original claims 8 and 15, but original claim 1, a method claim, did not recite either a "computer" or a "digital data processing device".

[3] There is another form of computer, an analog computer, which has been known for some time but which is rarely used. However, analog computers do not employ programming languages such as JAVA or other constructs of the type disclosed in the specification.

Regarding the phrase "digital data object", as explained above, the invention is practiced using a digital computer (a digital data processing device), and therefore any data processed by the digital computer is necessarily ***digital data***. The specification repeatedly uses the term "array object" in the context of data which is processed by a JAVA virtual machine, i.e., code executing on the digital computer. This array object is ***digital data*** processed by the computer, and would be so understood by those of skill in the art. Furthermore, the specification discloses that an array object (digital data) is allocated and stored in memory of the computer (see, e.g., p. 10, lines 1-3, lines 22-25, etc.) The specification therefore sufficiently discloses an "array object" which is a "digital data object".

Prior Art Rejections

Independent claims 1, 3, 8, 10, 15 and 17 have been amended to clarify certain aspects of the present invention. In particular, the claims have been amended to clarify that the flags for the multidimensional array object are flags maintained at execution time, which dynamically alterable according to execution state. As amended, the claims are patentable over the cited art.

Applicants' invention relates to the optimization of computer processes performed on certain arrays, particularly in object-oriented languages, of which JAVA is an example. Generally, it is possible to realize substantial execution performance efficiencies if it can be known in advance that the array will conform to a some regular structure, e.g., the array occupies a contiguous area of memory. However, it is often difficult to know this at compile time. Several prior art approaches are discussed in applicant's background, but none is completely satisfactory. In accordance with applicants' invention, dynamic flags are maintained during execution to indicate whether a multidimensional array has the requisite regular structure necessary for performing a code optimization. These flags are subject to change according to the execution state of the program. E.g., in the preferred embodiment, where it can not be definitely determined based on source code alone whether the array will have sufficient regular structure, the flags are initially set to indicate that the requisite regular structure is present, i.e. optimization is possible. If thereafter, some event occurs during execution to change the characteristics of the array so that it is no longer amenable to optimization, the corresponding flags are reset, so that a more general case, but less optimized, code is executed. The optimized and non-optimized code could be generated in advance by a static compiler, or could be generated by a "Just In Time" (JIT) compiler.

Cierniak, the primary reference cited by the Examiner, also involves in a general sense the optimization of executable code, particularly in JAVA. *Cierniak* further discloses that their technique is applicable to optimization of certain code for multi-dimensional array references. However, here the similarity ends. *Cierniak* discloses a technique based on analysis of the source code. Specifically, according to *Cierniak*, the context of source code is often lost when source is translated into JAVA bytecodes, thus making it difficult to optimize certain operations. *Cierniak* discloses the use of another intermediate representation (called "JavaIR"), which contains some of the contextual information and can be recovered from the bytecode representation. The JavaIR representation is used as an analytical tool to support optimization of executable code generated by the compiler.

The fundamental difference between applicants' claimed invention and *Cierniak* is that applicants' invention uses the dynamic execution state to determine an appropriate optimization. The original claims recited "managing" an array of flags, which in the context of the specification clearly meant managing at execution time. However, in order to clarify this important aspect of applicant's invention, applicants have amended the independent claims herein to make this more explicit. Representative amended claim 1 recites:

1. A method for processing a multidimensional array object comprising array objects, said multidimensional array object and said array objects being digital data objects storable in addressable data storage locations of a digital data processing device, said method comprising the steps of:

managing flags for said multidimensional array object at execution time of a process for elements of said multidimensional array object, *said flags being dynamically alterable at execution time to represent whether it is possible to optimize said process for elements of said multidimensional array object according to a current execution state of said process*, said process being a defined set of instructions executable by said digital data processing device; and

executing a machine code performing said process, said machine code being selected from among a plurality of machine codes performing said process according to a state of said flags. (emphasis added)

Independent claims 8 and 15 contain analogous recitations. Independent claims 3, 10 and 17 recite in somewhat different language that the flags are managed at execution time and inverted when the condition is no longer met by execution of the process, i.e, these claims also recite, although in different terms, that the flags are something dynamic.

As disclosed and claimed by applicants, optimization is selected during execution according to the dynamic state of flags, which can actually change during execution, so that it may be possible at one point to execute an optimized code, and later be necessary to use a non-optimized code to execute the same source code statement. *Cierniak* uses only the source code analysis to determine a level of optimization, and does not teach or suggest applicants' claimed technique.

It is indeed true that *Cierniak* discloses just-in-time (JIT) compilation, which is well-known in JAVA. But *Cierniak*'s JIT compilation analysis is no different from their static compilation analysis. Neither one uses dynamic flags associated with a multidimensional array, which can change with execution state, to determine whether optimized code is possible.

The advantage of applicants' approach is that it facilitates a more aggressive optimization. Using applicants' technique, it is possible to make optimistic assumption about optimization (which will in most cases turn out to be true), because the dynamic flags will prevent optimized code from being used in those relatively few instances where the optimistic assumptions are not true. *Cierniak* can not make optimistic assumptions, because in the few cases where the optimistic assumption turns out to be false, the program will behave in an unpredictable fashion. *Cierniak* must therefore make worst case assumptions, given the uncertainty of its source code analysis.

The secondary references cited herein fail to teach or disclose critical aspects of applicants invention, even assuming they are combined with *Cierniak*. *Hastings* discloses that flags can be associated with elements in data arrays, but these flags are being used for a completely different purpose. There is no teaching or suggestion to use dynamic flags which indicate whether it is possible to optimize code for processing an array. *Wolczko* is cited to show writing to an array using a write-barrier instruction, but again the context is completely different, and there is no teaching or suggestion of the key element of using dynamic flags to indicate whether code optimizations are possible.

In view of the foregoing, applicants submit that the claims are now in condition for allowance and respectfully request reconsideration and allowance of all claims. In addition, the

Examiner is encouraged to contact applicants' attorney by telephone if there are outstanding issues left to be resolved to place this case in condition for allowance.

Respectfully submitted,

TATSUSHI INAGAKI, et al.

By: [illegible]
Roy W. Truelson
Registration No. 34,265

Telephone: (507) 289-6256